Agent-oriented Semantic Discovery and Matchmaking of Web Services

Ivan Mećar[1], Alisa Devlić[1], Krunoslav Tržec[2]

[1]University of Zagreb
Faculty of Electrical Engineering and Computing
Department of Telecommunications
Unska 3, HR-10000 Zagreb, CROATIA
E-mail: {ivan.mecar, alisa.devlic}@fer.hr

[2]Ericsson Nikola Tesla
R&D Center
Krapinska 45, HR-10000 Zagreb, CROATIA
E-mail: krunoslav.trzec@ericsson.com

Abstract — **The article deals with implementation issues of semantic matchmaking of Web services using intelligent middle agents in an electronic market (e-market). The idea of comparison of Web services according to their semantic descriptions is used to find the most appropriate service that meets user preferences. Comprehensive description of not only the syntax, but also semantic meaning of the service is used to find the most suitable match. Matchmaking procedure is based on DAML-S (DARPA Agent Markup Language for Services) ontology, which contains required semantic information for discovery and comparison as well as execution and monitoring of Web services. In order to find a service that satisfies user requirements, the intelligent software agent semantically compares descriptions of requested and advertised Web services using matchmaking algorithm and automated reasoner based on description logics. This way user will be provided with the most appropriate service that matches his requirements.**

Keywords-Semantic matchmaking; intelligent software agents; DAML+OIL; DAML-S; Semantic Web

I. INTRODUCTION

The Semantic Web will enable software agents to autonomously find eligible Web services [1]. This idea can be realized by providing semantic descriptions of advertised and requested services. By adding additional metadata in order to describe existing Web services, agents will be able to recognize and understand the semantics of Web service capabilities. Consequently, the vison of Semantic Web will enable further step to complete automation of the trading activites in an electronic market of Web services. Intelligent software agent, developed and presented in this article represents a middle agent in the e-market that assists trader agents in semantic discovery and comparison of Web services offered by service provider that participate in the e-market. Because of matchmaking capabilities, this intelligent software agent is called matchmaking agent.

Using implemented matchmaking agent, it is possible to semantically discover required Web services. Service providers have to describe offered Web services with DAML-S service ontology in order to be advertiesed in the e-market [2]. Buyers also have to describe requested Web services using DAML-S ontology. The middle agent uses existing matchmaking algorithm to compare requested service with advertised services [3]. Consequently, the requested service isn't an actual service, but an ideal abstraction of a Web service. If it happens that the advertised service is exactly what the requester was looking for, then the advertised service completely matches the requested service. The contribution of this work is the implementation of the intelligent middle agent that uses the matchmaking algorithm to determine the semantic level of matching requested and advertised Web services. The elements of the DAML-S service descriptions are considered and matched individually. With this ranking it is possible to select appropriate advertised service among large set of results. The article is organized as follows: First, we discuss the existing Web service technologies and their shortcomings in the Semantic Web vision and therefore introduce the need for the shared ontologies; Next, we describe the Semantic Web languages and give a description of DAML-S ontology; Finally, we briefly explain the matchmaking algorithm and give the implementation overview of the matchmaking agent and its software components.

II. EXISTING WEB SERVICE TECHNOLOGIES

Web Service Description Language (WSDL) is a language that provides a communication level description for a Web service [1]. A WSDL document is, basically, the XML (Extensible Markup Language) document specifying the location and operations of the service and how to access it. WSDL files are XML files with no explicit semantic data.

Universal Description, Discovery and Integration (UDDI) is the existing specification for a repository to find Web services in the Internet [1]. The repository is a business registry that enables businesses to locate on quick, dynamic and easy way the other business partners. Businesses register their services with UDDI, and the UDDI business registry uses standard industry taxonomies, or classification schemes to categorize business, services and service types (so called Yellow pages). Typically, when a business registers a Web service, besides the usual information, it also stores a WSDL description of the service, or a reference to the WSDL document. This specification then enables the user to easily connect to the Web service. Basically, UDDI allows only keyword search based on the name of businesses and services.

However, WSDL does not support semantic description of services. For example, it does not provide the definition of logical constraints between its input and output parameters. Furthermore, UDDI does not represent service capabilities so the matchmaking can only be done by string matching on the defined attributes such as name or address of service provider.

III. Semantic Descriptions of Web Services

Today, Web is a remarkable source of information. This information is mostly structured in the form of human understandable Web pages (i.e. plain text), which makes this information unrecognizable for computer programs. The concept of the Semantic Web allows for the unambiguous interpretation of Web resources and content through the use of shared Web ontologies. DAML-S is an example of ontology that provides a semantic markup for Web services, offering an automation of business processes (e.g., automatic discovery of a Web service). The ontologies represent shared descriptions that consist of concepts as well as relationships between these concepts that are important for a domain of interest. These predefined ontologies allow software agents to interpret the meaning of Web resources. Ontologies can refer to other ontologies, providing domain-dependent terminologies that describe some concepts and relationships in more detail. Ontologies are explicit semantic models, which include taxonomies of terms and semantic relations that help in interpreting described knowledge [1].

DAML-S ontology provides semantic description of Web service capabilites [2]. Web services that come along with semantic information therefore become meaningful to software agents. DAML-S integrates rich class representations with a process model designed to capture not only the control and data flow of Web services but also their real world side effects (preconditions and effects). Its well-defined semantics allows automated service matchmaking in the e-market with a known outcome using powerful reasoning techniques. Consequently, using appropriate software components, agents in the electronic market are able to autonomously decide whether a particular Web service satisfies certain requirements or not, which results in even greater automation of the trading processes in the e-market [2].

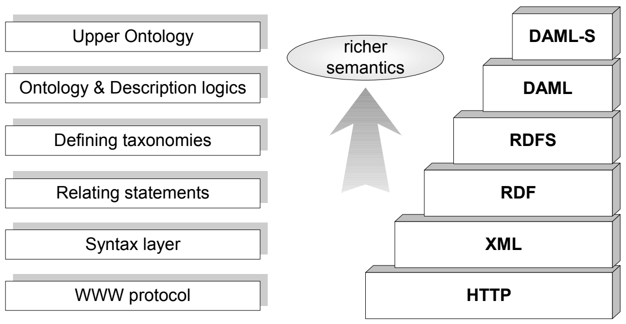


Figure 1. The Semantic Web languages

DAML-S makes use of DAML+OIL (DARPA Agent Markup Language + Ontology Inference Layer) that is an Artificial Intelligence (AI) inspired description logic (DL)-based language [4]. Description logics is a formalism that we can use for knowledge representation and reasoning. It gives us ability to find implicit consequences of explicitly represented knowledge. DAML+OIL is suitable for knowledge representation of Web service capabilites as:

- It provides a reasonable level of flexibility and extensiveness while keeping a nice balance between expressiveness and decidability. The support for types greatly enhances the expressiveness and modularity of the descriptions.
- DAML+OIL offers support for ontologies. It has been already integrated with tools which make the generation of new ontologies for service description much easier.
- It is a good candidate for expressing service descriptions that will be subject to the matchmaking operations. It is shown that all needed matchmaking procedures can be expressed in terms of the description logic subsumption operation [3]. Moreover, mature software components exist (so called DL reasoners) that can efficiently perform the subsumption operation on DAML+OIL descriptions.
- DAML+OIL offers support for expressing constraints, while still maintaining decidability.

Figure 1 depicts the Semantic Web languages. It can be seen that DAML-S is built on the top of DAML+OIL ontology language. RDF (Resource Description Framework) is a basic data model for writing statements about Web resources. The RDF does not relay on XML. However, it has an XML-based syntax. RDFS (RDF Schema) provides modeling primitives for organizing Web resources into hierarchies. Key primitives are classes and properties, subclass and subproperty relationships, and domain and range restrictions. RDF Schema can be viewed as a primitive language for writing

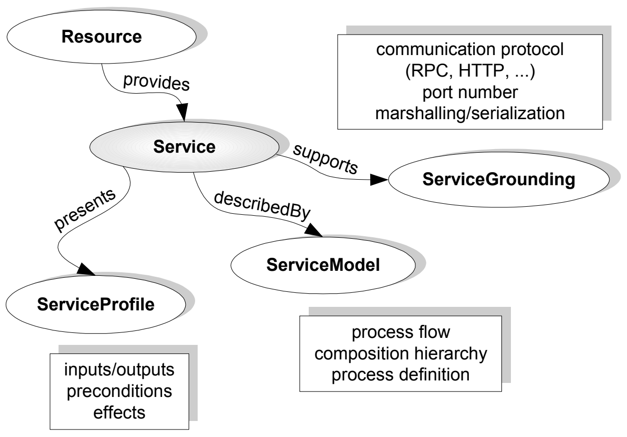


Figure 2. The structure of DAML-S ontology

ontologies based on RDF data model. If we need more expressive language for ontologies, than we can use DAML+OIL language based on RDFS.

IV. Structure of DAML-S Ontology

DAML-S ontology represents an upper ontology for describing Web services. It is conceptually divided into three subontologies: Service Profile (specifying what a service does), Service Model (specifying how the service works), and Service Grounding (specifying how the service is implemented) [1]. Structure of DAML-S ontology is made of four basic classes and every class is aimed to enable execution of one of main Web service tasks. This structure is represented on Fig. 2. The class `Service` provides an entry point for any Web Service description. It is an abstraction of real service. Exactly one instance of `Service` will exist for each published Web service. `Service` has three properties and the range of each property is another class. Each of that class provides essential type of knowledge about particular service.

The `ServiceProfile` class provides a precise description of the functionality of service in order to enable software agent to make decision whether the service satisfy its demands. Each method that a service provides is described in a class `Profile`. A service profile is actually presented through the class `Profile`, which is a direct subclass of the class `ServiceProfile`. The `ServiceModel` class describes what happens when the service is executed. The exact functionality is presented through a process model (i.e. through Process ontology and Process Control ontology). Finnaly, the `ServiceGrounding` class specifies how an agent can access a service. "Grounding" specifies communication protocol, message format, port number, etc.

Since the matchmaking agent is created to find the most appropriate advertised Web service, it will use the `Profile` class of both requested and advertised DAML-S service descriptions. Matching procedure implemented in matchmaking agent will efficiently find the semantic similarity between `Profile` class of requested service and `Profile` class of advertised service. The profile of the service (`Profile` class) contains three basic types of information used in matchmaking process that occurs in an electronic service market.

First type of information includes textual description and contact information (like `name`, `title`, `phone`, `email`, etc.), which is mainly intended for human users. The second type of information provides a functional description of the service. This is the main part of DAML-S document used by the matchmaking agent. Functional description is expressed with `input` parameters and `output` parameters generated by the service. Additionally, the functional description contains two sets of conditions: `preconditions` and `effects`. `Preconditions` have to hold before service can be properly executed, and `effects` have to hold after successful execution of the service. These four functional descriptions are also named IOPE. Every IOPE parameter has his own three properties: `parameterName`, `restrictedTo` and `refersTo`. Finally, the third type of information comes with the set of additional properties of class `Profile` that are used to describe additional features of the service. The ranges of these additional properties are classes `QualityRating`, `ServiceParameter` and `ServiceCategory`.

The `ServiceProfile` is the central entity that enables agents for seeking and comparing service advertisements against users requests. However, in some cases, intelligent agents might find `ServiceModel` useful for details about particular service, because `ServiceModel` describes the dynamic behavior of the service and gives more details about its functions. These details could be found inspecting the service Process Model.

V. Matchmaking Algorithm

Matchmaking algorithm is a procedure that will assist the software agent in selecting the most suitable Web Service for the given preferences. The algorithm is based on the available semantic information encoded in DAML-S. More specifically, it will match (compare) advertised service parameters with requested capabilites (parameters). This match will result with some matching degree, a ranking result. Such ranking will eventually become necessary since it is highly unlikely that there will always be the service that offers exactly the specified functionality from requester's point of view. Based on these rankings, users (or software agents that act on behalf of users) can decide if they want to make the use of a Web Service that does not exactly match the desired functionality or can use the service that match this functionality to some extent.

The main characteristic of the algorithm is splitting matchmaking procedure into several parts: input matching, output matching, profile matching and user-defined matching. The procedure is logically divided into four stages, each independent on other three. The final result will be based on the results of each matchmaking stage. Division of algorithm is very logical and fully compatible with DAML-S Profile. We used the algorithm that inspects only `ServiceProfile` to get needed information about the particular Web service.

1) *INPUT PARAMETERS MATCHING*

In input parameters matching, the algorithm attempts to determine how good the inputs of the advertised service correspond to the inputs of the requested service. For each input of the advertised service, the algorithm tries to find a match with an input of the requested service using `property_match` and `type_match` functionality. The best match will constitute a pair. This is made for all parameters of advertised and requested inputs. When looking at matching results for all input pairs, the final result represents worst-case scenario. The input parameters degree of match can be: FAIL (0), UNCLASSIFIED (1), SUBPROPERTY (2), TYPE_INVERT (3), TYPE_SUBSUMES (4), and MATCH (5).

2) *OUTPUT PARAMETERS MATCHING*

In output parameters matching, the algorithm attempts to determine how good the outputs of the advertised service correspond to the outputs of the requested service. For each output of the requested service, the algorithm tries to find the matching output parameter of the advertised service which will result in the highest rank. This "one shot" matching is again composed of `property_match` and `type_match` functionality. When looking at all output pairs being matched, the final result is worst-case scenario. The output parameters degree of match can be: FAIL (0), PARTIAL_FAIL (1), UNCLASSIFIED (2), SUBPROPERTY (3), TYPE_INVERT (4), TYPE_SUBSUMES (5), and MATCH (6).

3) *PROFILE MATCHING*

Profile matching is different than the IOPE matching explained above. It attempts to determine how good the service category of the advertised service fits into service category of the requested service. This matching is based on possibility of classifying the class `Profile` into subclasses/subcategories. Consequently, Profile matching is based on concept matching, i.e. matching between classes. Possible degrees of profile matching are: FAIL (0), UNCLASSIFIED (1), SUBSUMES (2), and MATCH (3).

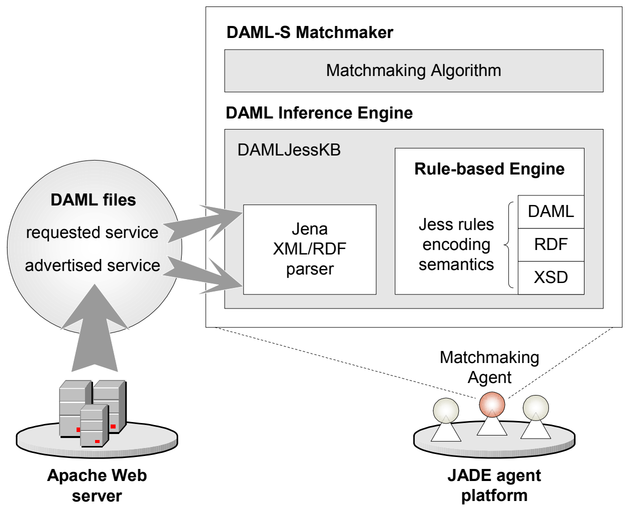


Figure 3. Software components required for semantic matchmaking

4) *USER-DEFINED MATCHING*

This fourth and last stage of matchmaking algorithm allows the entity that manages the algorithm to define additional matching functions with plug-in. There can be many plug-in matching types. If only one plug-in match fails, user-defined matching fails too. Final result of this matching can be either TRUE or FALSE. This matching gives us possibility to exploite Quality of Service (QoS) through class `QualityRating`, the property of `Profile` class.

5) *FINAL MATCHING RESULT*

Final matching result can be either MATCH or FAIL, and represents "logical AND" between all four stages of matching procedure.

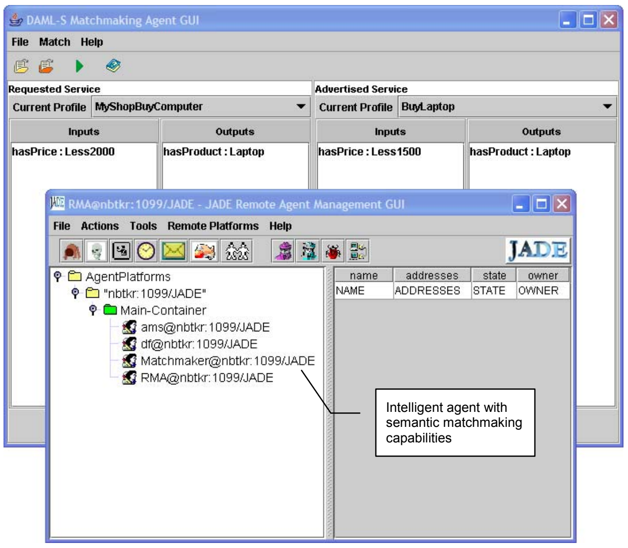
VI. MATCHMAKING AGENT COMPONENTS

Implemented matchmaking agent uses two components to realize semantic matchmaking process of required and advertised Web services. These components are:

- DAML Inference Engine
- DAML-S Matchmaker

These components are necessary to accomplish complex reasoning tasks, including Java understandable interpretation of requester's and advertiser's service written in DAML-S.

The DAML Inference Engine component is used to transform DAML files in the form appropriate for the DAML-S Matchmaker component that, using Matchmaking Algorithm, semantically compares transformed DAML files and calculate degree of similarity between Web services. The DAML Inference Engine represents off-the-shelf DL reasoner for ontologies written in DAML+OIL, named DAMLJessKB [5], which uses Jena API to parse DAML-S descriptions,


DAML-S Matchmaking Agent GUI
File Match Help
RMA@nbtkr:1099/JADE - JADE Remote Agent Management GUI
File Actions Tools Remote Platforms Help
JADE
Intelligent agent with semantic matchmaking capabilities

Requested Service — Current Profile: MyShopBuyComputer

Inputs	Outputs
hasPrice : Less2000	hasProduct : Laptop

Advertised Service — Current Profile: BuyLaptop

Inputs	Outputs
hasPrice : Less1500	hasProduct : Laptop

AgentPlatforms
"nbtkr:1099/JADE"
Main-Container
ams@nbtkr:1099/JADE
df@nbtkr:1099/JADE
Matchmaker@nbtkr:1099/JADE
RMA@nbtkr:1099/JADE

name	addresses	state	owner
NAME	ADDRESSES	STATE	OWNER

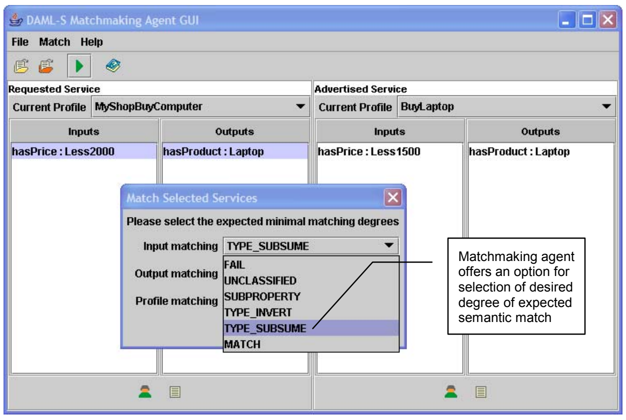
Figure 4. GUI of JADE agent platform running matchmaking agent

fetched from Apache Web server, into RDF subject-verb-object (SVO) triples. DAMLJessKB also uses Jess (Java Expert System Shell), a rule-based engine that can be used for creation of agent knowledge base (KB) populated with facts and rules. Jess uses Rete algorithm (pattern matching mechanism) to process facts and deduce new information according to rules. It is used to enable the intelligent agent to process SVO triples, represented as Jess facts, according to Jess rules that represent DAML+OIL axioms. Consequently, the intelligent agents can understand semantics of DAML-S descriptions using the DAMLJessKB component and use the DAML-S Matchmaker component to semantically match requested and advertised services using all available semantic information loaded into the KB.

VII. Matchmaking Agent Implementation

The matchmaking agent is built and deployed in JADE agent platform [2], as it is shown in Fig. 4. During the agent development, component-based approach is adopted. In order to perform semantic service matchmaking using the matchmaking algorithm, the intelligent agent fetches DAML-S documents from the Apache Web server that is used as repository of DAML-S service descriptions.

After reaching DAML-S descriptions and downloading them, the matchmaking agent offers Graphical User Interface (GUI) that enables the selection of satisfying degree of input, output and profile matching stages. The GUI for semantic matchmaking is shown in Fig. 5. It enables user-friendly determination of the adequate matchmaking degree between requested and advertised service descriptions. For example, we don't need to expect that matchmaking algorithm will end with highest degree of success in all its four stages. The lowest degree that must be satisfied is one selected by the user.


DAML-S Matchmaking Agent GUI
File Match Help
Match Selected Services
FAIL
UNCLASSIFIED
SUBPROPERTY
TYPE_INVERT
TYPE_SUBSUME
MATCH
Matchmaking agent offers an option for selection of desired degree of expected semantic match

Requested Service — Current Profile: MyShopBuyComputer

Inputs	Outputs
hasPrice : Less2000	hasProduct : Laptop

Advertised Service — Current Profile: BuyLaptop

Inputs	Outputs
hasPrice : Less1500	hasProduct : Laptop

Please select the expected minimal matching degrees
Input matching: TYPE_SUBSUME
Output matching
Profile matching

Figure 5. GUI for semantic matchmaking

These offered degrees were already listed in this paper when the stages of the matchmaking algorithm were discussed (input parameter matching, output parameter matching, profile matching and user-defined matching). After the matchmaking process, the message with the result of every stage (MATCH or FAIL), as well as final result (MATCH or FAIL) appears.

VIII. Conclusion

The implemented intelligent matchmaking agent provides the automation of semantic Web service discovery and matchmaking in an electronic market for Web services. The semantic matchmaking is enabled by the use of DAML-S ontology for describing Web services, which has well-defined semantics, based on descripton logics formalism.

The matchmaking agent is built using component-based approach. It contains the off-the-shelf component for reasoning with DAML+OIL ontologies as well as matchmaking algorithms for DAML-S service descriptions that enable finding the most suitable service according to user preferances, resulting in enhanced utilization of Web services in the e-market.

References

[1] G. Antoniou and F. van Harmelen, *A Semantic Web Primer*, MIT Press, Cambridge, Massachusetts, US, 2004.

[2] K. Tržec, A. Devlić, G. Ježić, M. Kušek, and S. Dešić, "Semantic Matchmaking of Advanced Personalized Mobile Services using Intelligent Agents", *Proceedings of the 12th International Conference on Software, Telecommunications and Computer Networks (SoftCOM 2004)*, pp. 387-391, Split, Dubrovnik, Croatia, Venice, Italy, 2004.

[3] S. Tang, "Matching of Web Services Specifications using DAML-S descriptions", Technical University of Berlin, Berlin, Germany, 2004.

[4] D. L. McGuinness, R. Fikes, J. Hendler, and L. A. Stein, "DAML+OIL: An Ontology Language for the Semantic Web", IEEE Intelligent Systems, vol. 17, no. 5, pp. 72-80, 2002.

[5] J. Kopena and W. C. Regli: "DAMLJessKB: A Tool for Reasoning with the Semantic Web", *IEEE Intelligent Systems*, vol. 18, no. 3, pp. 74-77, 2003.